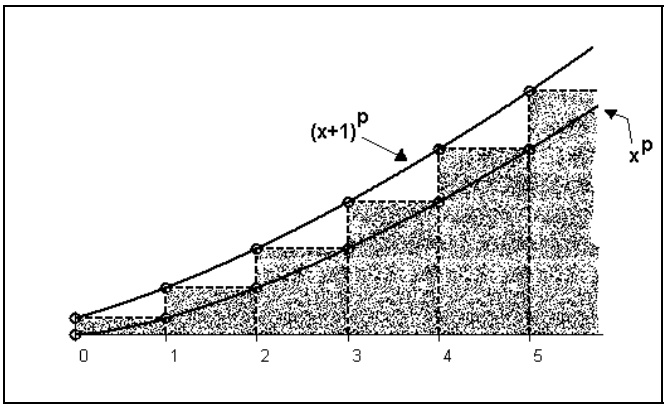
Estimating Operation Counts for Algorithms

Most analyses of computation complexity of linear algebra algorithms center on counting the number of operations required to implement an algorithm. In most cases, this number is expressed as one or more finite sums of the general form

$$\sum_{k=1}^{n} k^p$$

While we can develop explicit formulas for the values of such sums when p is any integer, in most such cases, we really only need *estimates* that are accurate to within a few percent. It turns out we can find these estimates very simply, using only a little calculus.

Specifically, consider the following picture



It should be fairly clear that each of the shaded rectangles has a base of one, and that their heights are, respectively, 1^p , 2^p , 3^p , ..., and therefore, $\sum_{k=1}^{n} k^p$ will be exactly equal to the area under that part of the "staircase" lying between $x = 0$ and $x = n$. But, it should also be obvious that this staircase area must be greater than the area under the lower of the two curves, and less than the area above the higher of them, i.e.

$$\int_0^n x^p \, dx \leq \sum_{k=1}^{n} k^p \leq \int_0^n (x+1)^p \, dx$$

i.e.

$$\frac{n^{p+1}}{p+1} \leq \sum_{k=1}^{n} k^p \leq \frac{(n+1)^{p+1}}{p+1}$$

Furthermore, for "large" n, say $n > 100$, the upper and lower bounds here are actually within a couple of percent or less of each other, and therefore either could be used as an estimate to the value of the sum.